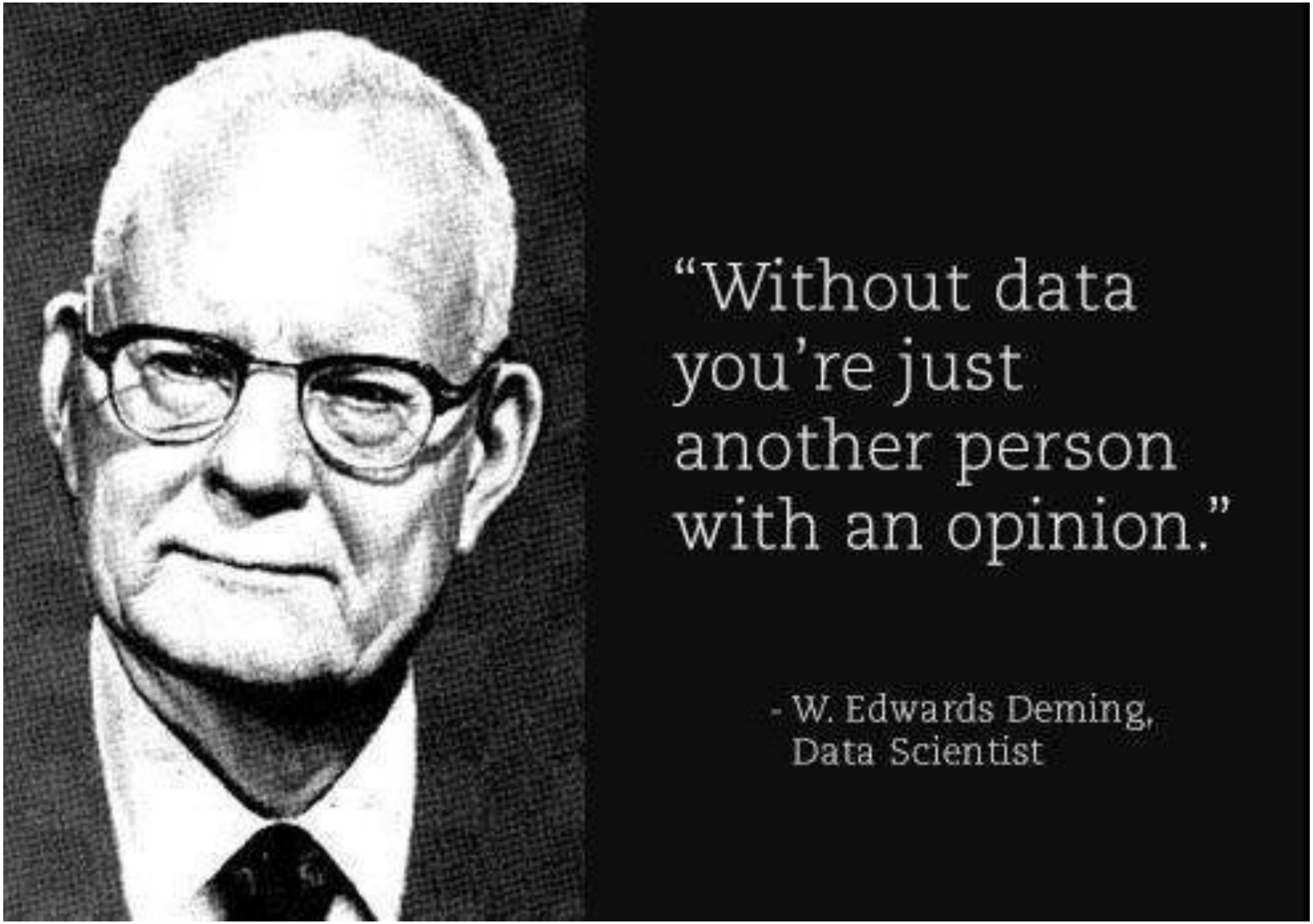
HoGent

BEDRIJF
EN
ORGANISATIE

Databases II

SQL: basic concepts revisited


“Without data you’re just another person with an opinion.”
- W. Edwards Deming, Data Scientist

Introduction

Overview

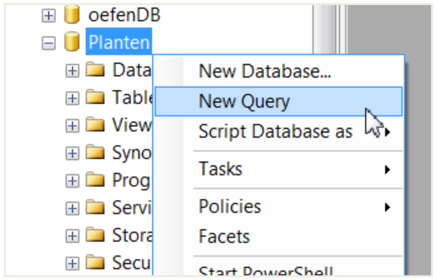
- **(Microsoft) SQL**
  - Working with 1 table: SELECT, Statististical functions, GROUP BY
  - Working with > 1 tables: JOIN, UNION, subquery's, correlated subquery's
  - Modifying data: insert, update, delete
  - Views

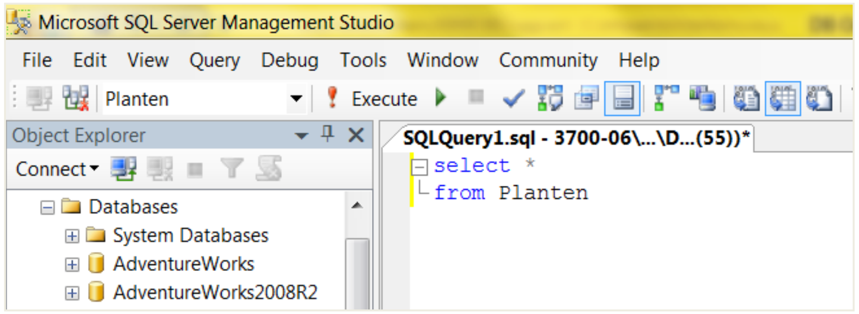
SQL Server

- SQL Server :
  - Management
    - Installation, configuration and security of SQL Server.
    - Database creation
    - Database management: backup, restore, ...
    - Use SQL Server Management Studio

Writing queries

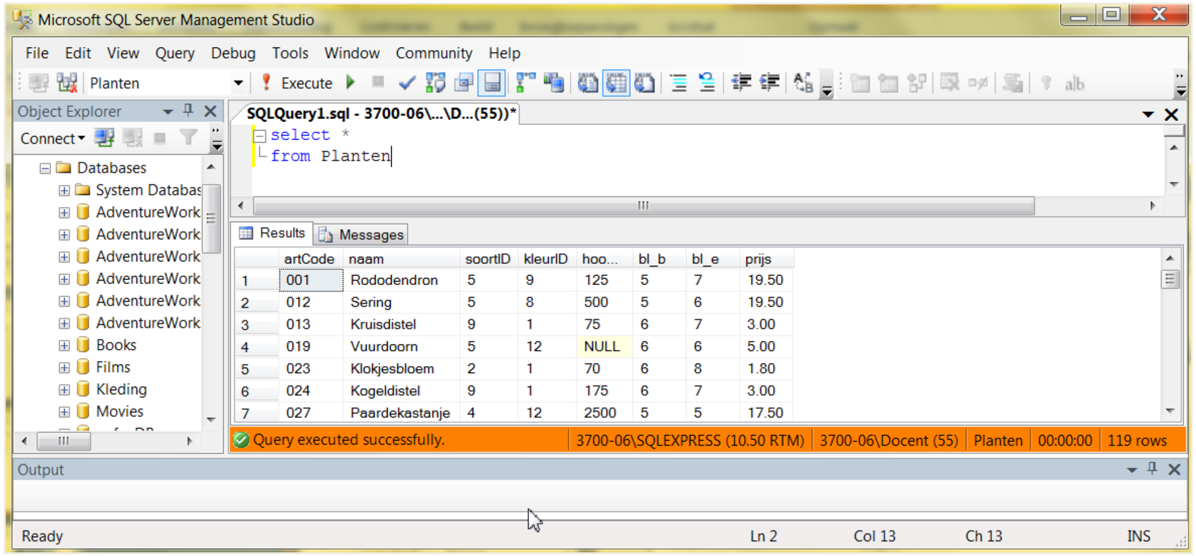
- Use SQL Server Management Studio





Microsoft SQL Server Management Studio
File Edit View Query Debug Tools Window Community Help
Planten
Execute
Object Explorer
Connect
Databases
System Databases
AdventureWorks
AdventureWorks2008R2
SQLQuery1.sql - 3700-06\...\D...(55))*

```
select *
from Planten
```



```
select *
from Planten
```

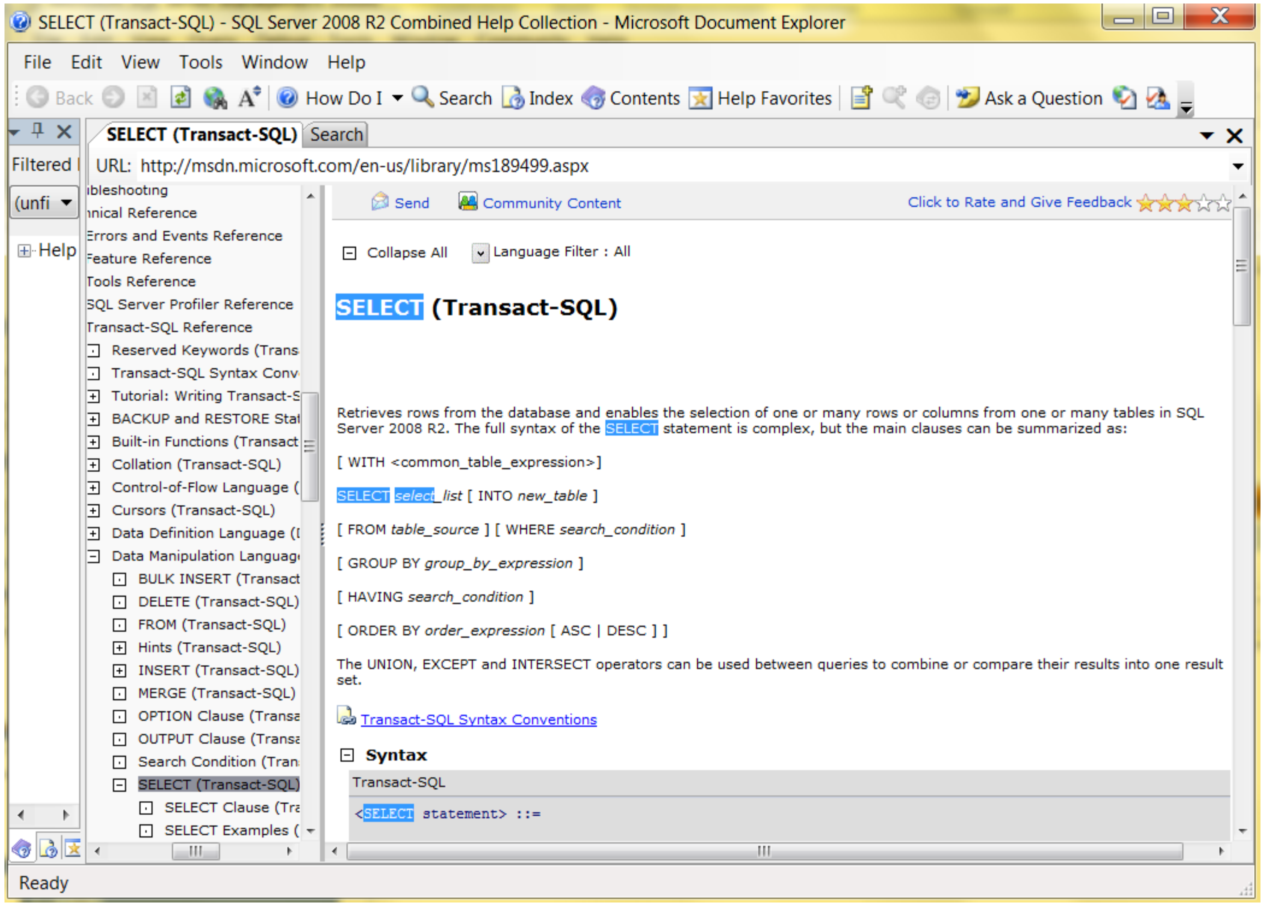
	artCode	naam	soortID	kleurID	hoo...	bl_b	bl_e	prijs
1	001	Rododendron	5	9	125	5	7	19.50
2	012	Sering	5	8	500	5	6	19.50
3	013	Kruisdistel	9	1	75	6	7	3.00
4	019	Vuurdoorn	5	12	NULL	6	6	5.00
5	023	Klokjesbloem	2	1	70	6	8	1.80
6	024	Kogeldistel	9	1	175	6	7	3.00
7	027	Paardekastanje	4	12	2500	5	5	17.50

Query executed successfully. | 3700-06\SQLEXPRESS (10.50 RTM) | 3700-06\Docent (55) | Planten | 00:00:00 | 119 rows

HoGent

help!

- The help menu offers online help about Microsoft SQL


SELECT (Transact-SQL) - SQL Server 2008 R2 Combined Help Collection - Microsoft Document Explorer
File Edit View Tools Window Help
Back How Do I Search Index Contents Help Favorites Ask a Question
SELECT (Transact-SQL) Search
Filtered
(unfi
Help
URL: http://msdn.microsoft.com/en-us/library/ms189499.aspx
Send Community Content Click to Rate and Give Feedback
Collapse All Language Filter : All
Ready

ubleshooting
nical Reference
Errors and Events Reference
Feature Reference
Tools Reference
SQL Server Profiler Reference
Transact-SQL Reference
Reserved Keywords (Trans
Transact-SQL Syntax Conv
Tutorial: Writing Transact-S
BACKUP and RESTORE Stat
Built-in Functions (Transact
Collation (Transact-SQL)
Control-of-Flow Language (
Cursors (Transact-SQL)
Data Definition Language (I
Data Manipulation Languag
BULK INSERT (Transact
DELETE (Transact-SQL)
FROM (Transact-SQL)
Hints (Transact-SQL)
INSERT (Transact-SQL)
MERGE (Transact-SQL)
OPTION Clause (Transa
OUTPUT Clause (Transa
Search Condition (Tran
SELECT (Transact-SQL)
SELECT Clause (Tra
SELECT Examples (

SELECT (Transact-SQL)

Retrieves rows from the database and enables the selection of one or many rows or columns from one or many tables in SQL Server 2008 R2. The full syntax of the SELECT statement is complex, but the main clauses can be summarized as:

[WITH <common_table_expression>]

SELECT *select_list* [INTO *new_table*]

[FROM *table_source*] [WHERE *search_condition*]

[GROUP BY *group_by_expression*]

[HAVING *search_condition*]

[ORDER BY *order_expression* [ASC | DESC]]

The UNION, EXCEPT and INTERSECT operators can be used between queries to combine or compare their results into one result set.

Transact-SQL Syntax Conventions

Syntax

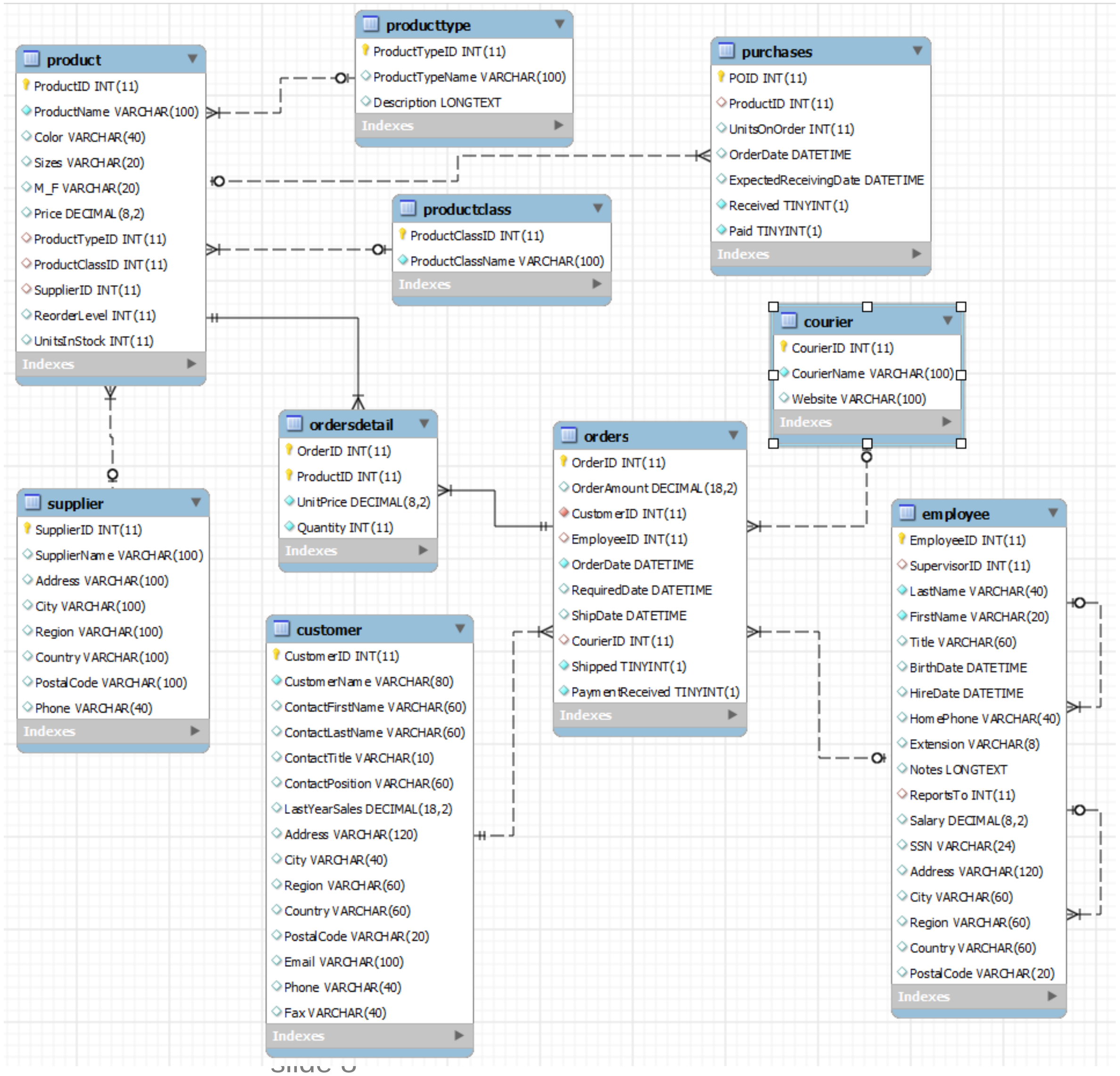
Transact-SQL

```
<SELECT statement> ::=
```

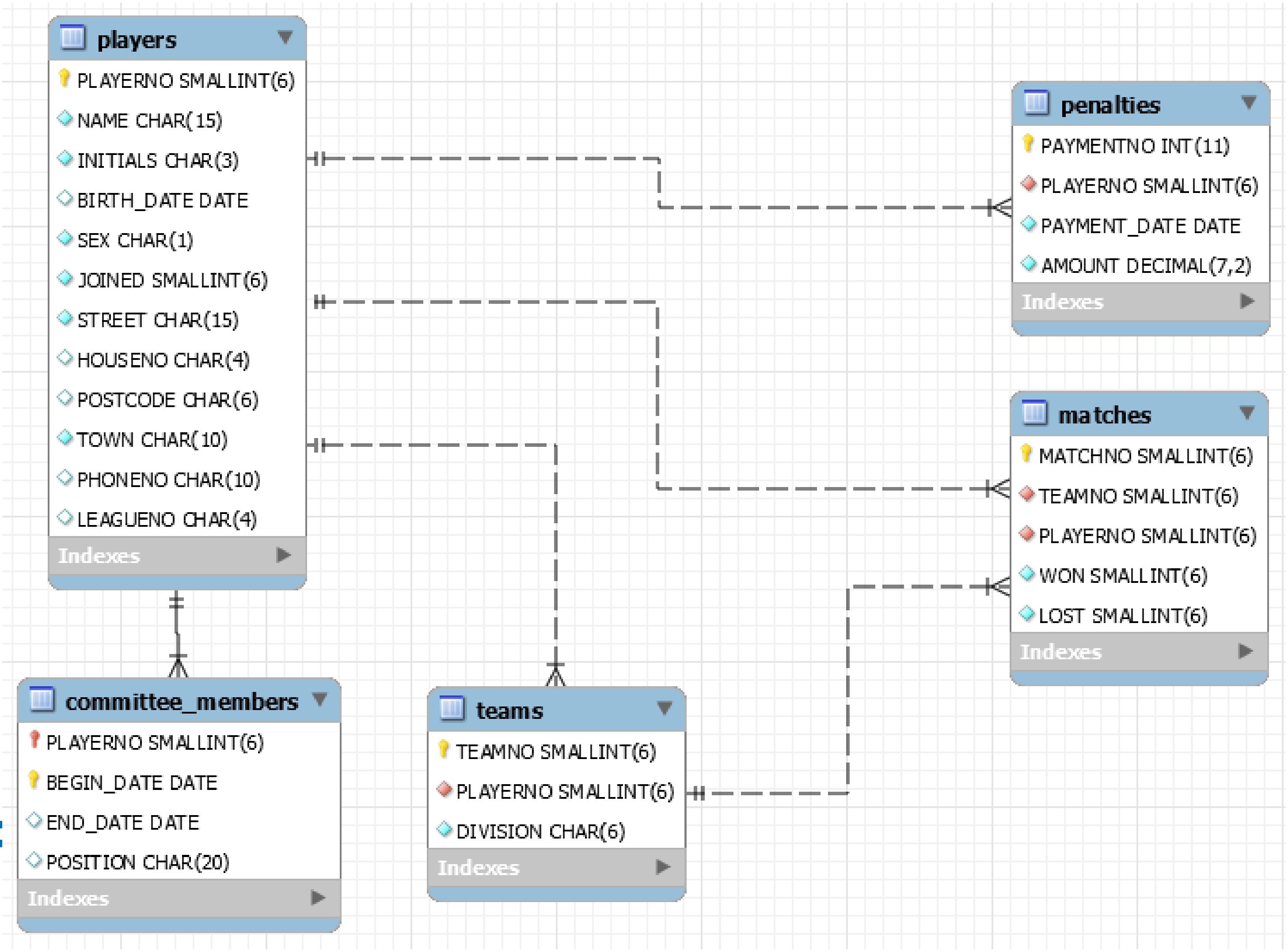
hoe schrijf ik ook al weer een correcte select opdracht? help!

HoGent

The DB 'xtreme': diagram

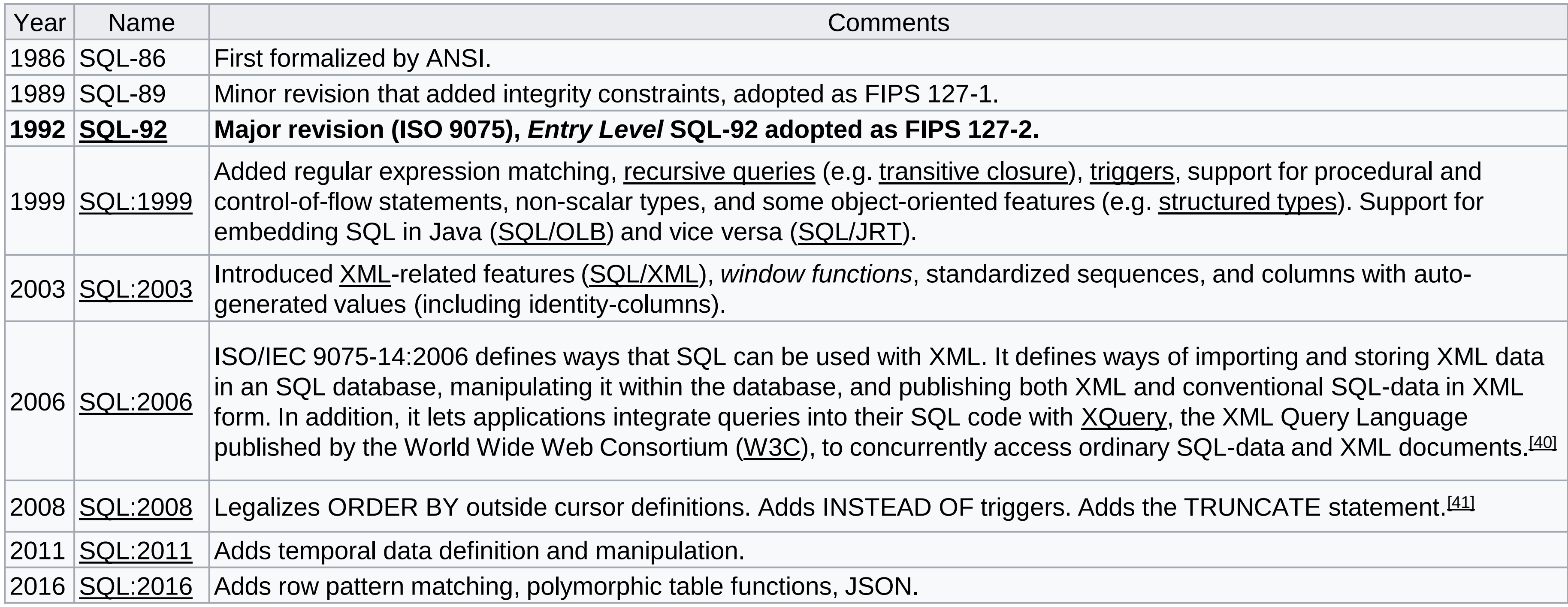


The DB 'tennis':



SQL - standards and dialects

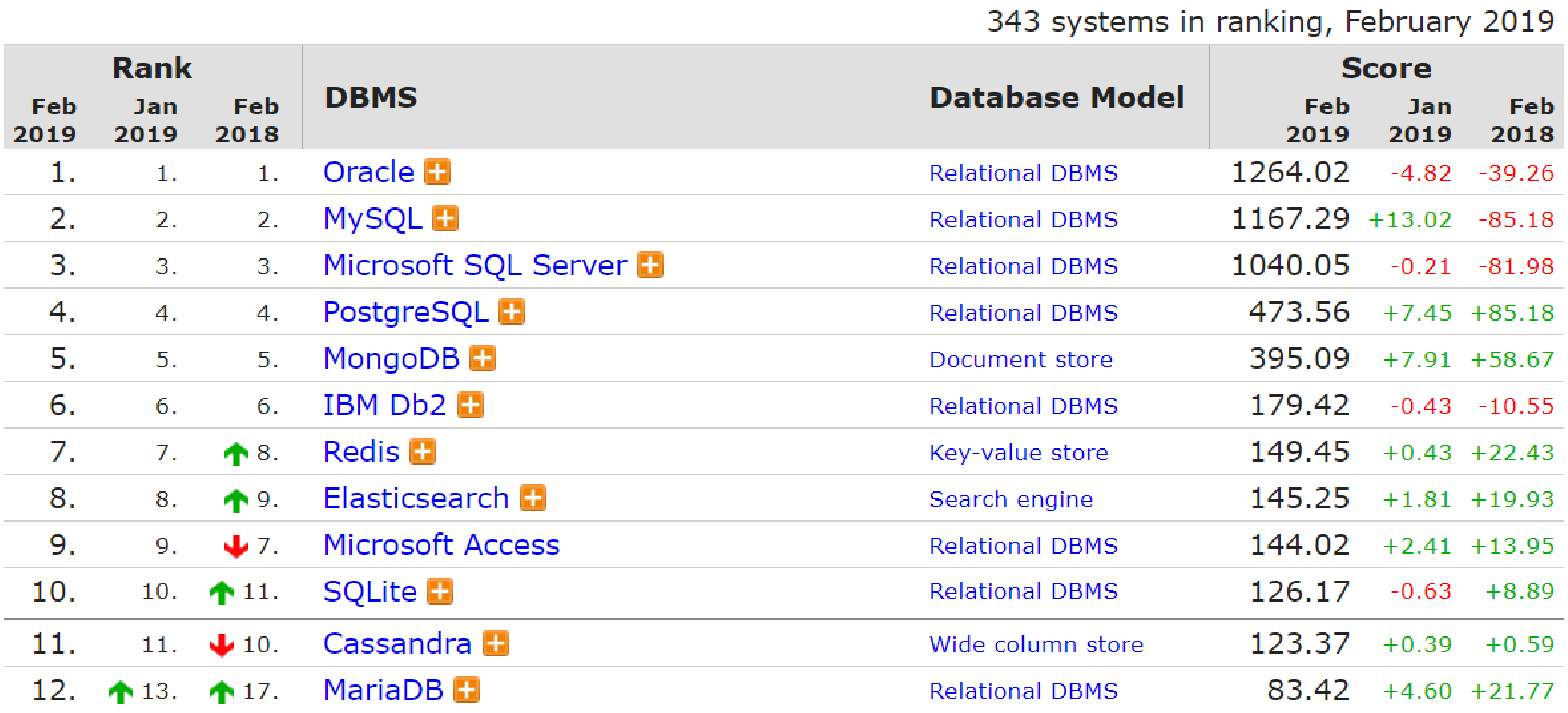
- Definition
  - Relational data language for relational database systems.
  - Non procedural language
- Standards



Year	Name	Comments
1986	SQL-86	First formalized by ANSI.
1989	SQL-89	Minor revision that added integrity constraints, adopted as FIPS 127-1.
1992	**SQL-92**	**Major revision (ISO 9075), *Entry Level* SQL-92 adopted as FIPS 127-2.**
1999	SQL:1999	Added regular expression matching, recursive queries (e.g. transitive closure), triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features (e.g. structured types). Support for embedding SQL in Java (SQL/OLB) and vice versa (SQL/JRT).
2003	SQL:2003	Introduced XML-related features (SQL/XML), *window functions*, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006	ISO/IEC 9075-14:2006 defines ways that SQL can be used with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database, and publishing both XML and conventional SQL-data in XML form. In addition, it lets applications integrate queries into their SQL code with XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.[40]
2008	SQL:2008	Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.[41]
2011	SQL:2011	Adds temporal data definition and manipulation.
2016	SQL:2016	Adds row pattern matching, polymorphic table functions, JSON.

HoGent

Why Microsoft SQL Server?



343 systems in ranking, February 2019

Rank			DBMS	Database Model	Score		
Feb 2019	Jan 2019	Feb 2018			Feb 2019	Jan 2019	Feb 2018
1.	1.	1.	Oracle	Relational DBMS	1264.02	-4.82	-39.26
2.	2.	2.	MySQL	Relational DBMS	1167.29	+13.02	-85.18
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1040.05	-0.21	-81.98
4.	4.	4.	PostgreSQL	Relational DBMS	473.56	+7.45	+85.18
5.	5.	5.	MongoDB	Document store	395.09	+7.91	+58.67
6.	6.	6.	IBM Db2	Relational DBMS	179.42	-0.43	-10.55
7.	7.	↑ 8.	Redis	Key-value store	149.45	+0.43	+22.43
8.	8.	↑ 9.	Elasticsearch	Search engine	145.25	+1.81	+19.93
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	144.02	+2.41	+13.95
10.	10.	↑ 11.	SQLite	Relational DBMS	126.17	-0.63	+8.89
11.	11.	↓ 10.	Cassandra	Wide column store	123.37	+0.39	+0.59
12.	↑ 13.	↑ 17.	MariaDB	Relational DBMS	83.42	+4.60	+21.77

Source: db-engince s.com

SQL - Overview

- SQL consists of 3 sub languages
  - **Data Definition Language (DDL)**
    - creation of a database, defining database objects (tables, stored procedures, views,…)
    - CREATE, ALTER, DROP
  - **Data Manipulation Language (DML)**
    - Querying and manipulating data in a database
    - SELECT, INSERT, UPDATE, DELETE
  - **Data Control Language (DCL)**
    - Data security and authorisation
    - GRANT, REVOKE, DENY
- Additional language elements: operators, functions, control of flow (dialects!)

SELECT

HoGent

DML – Consulting data

- Consulting one table
  - Basic form
  - SELECT clause
  - WHERE clause
  - Row formatting
  - Statistical functions
  - Grouping
- Consulting >1 table

Basic form of SELECT statement

- SELECT for consulting one table

```
SELECT [ALL | DISTINCT] {*|expression [, expression ...]}
FROM table name
[WHERE conditions(s)]
[GROUP BY column name [, column name ...]
[HAVING conditions(s)]
[ORDER BY {column name |seq nr}{ASC|DESC}[,...]
```

  - SELECT clause: specifies the columns to show in the ouput. DISTINCT filters out duplicate lines
  - FROM clause: table name
  - WHERE clause : filter condition on individual lines in the output
  - ORDER BY clause : sorting
  - GROUP BY : grouping of data
  - HAVING clause : filter condition on groups

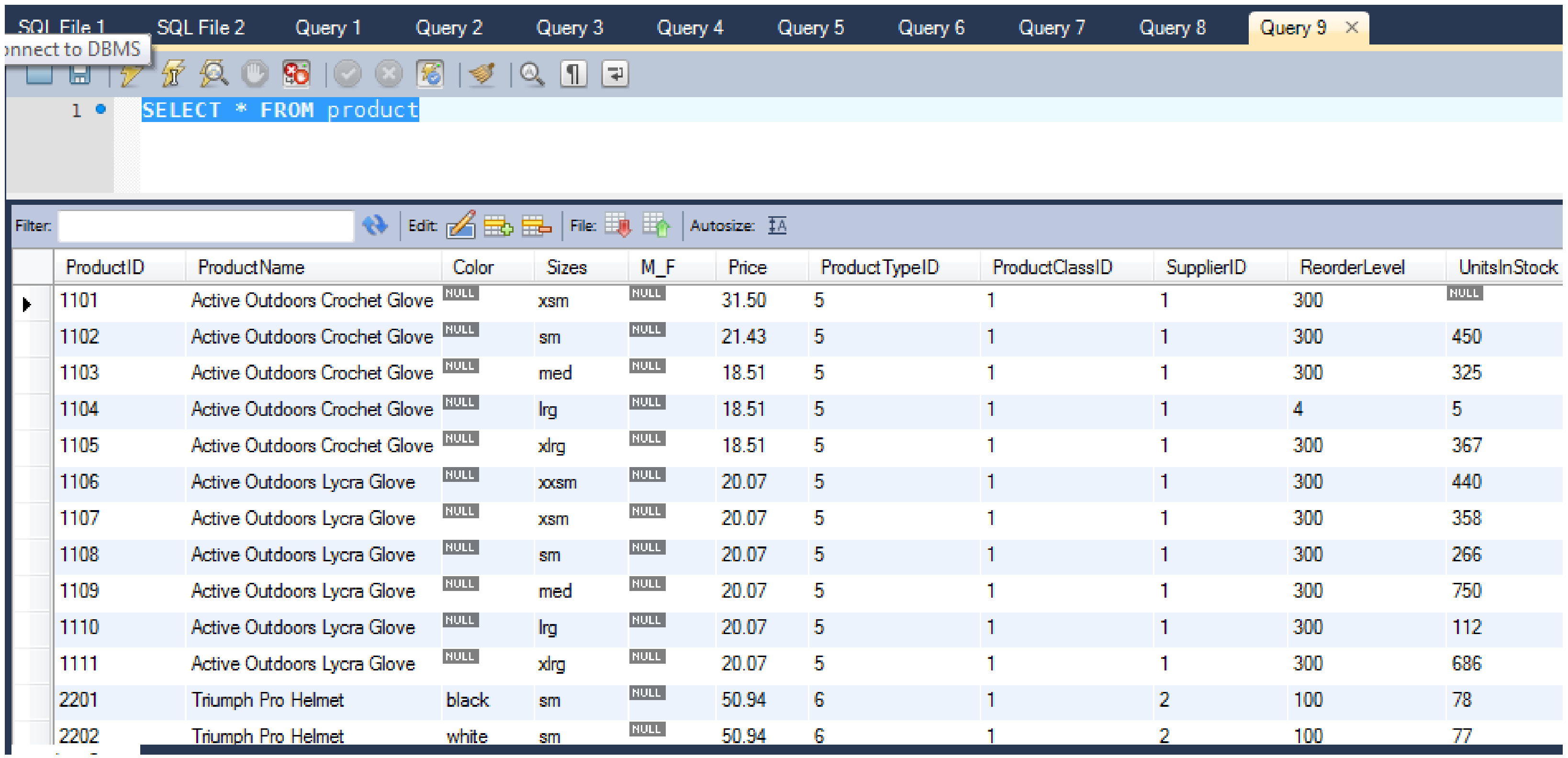
SELECT

- SELECT clause: specification of the columns
  – All columns from table: use *
    - SELECT *
  – Specific columns: use columns names or expression
    - SELECT column1 , column2, column3*column4, …

SELECT

- Example1 – show all data of all products

```
SELECT *
FROM product
```



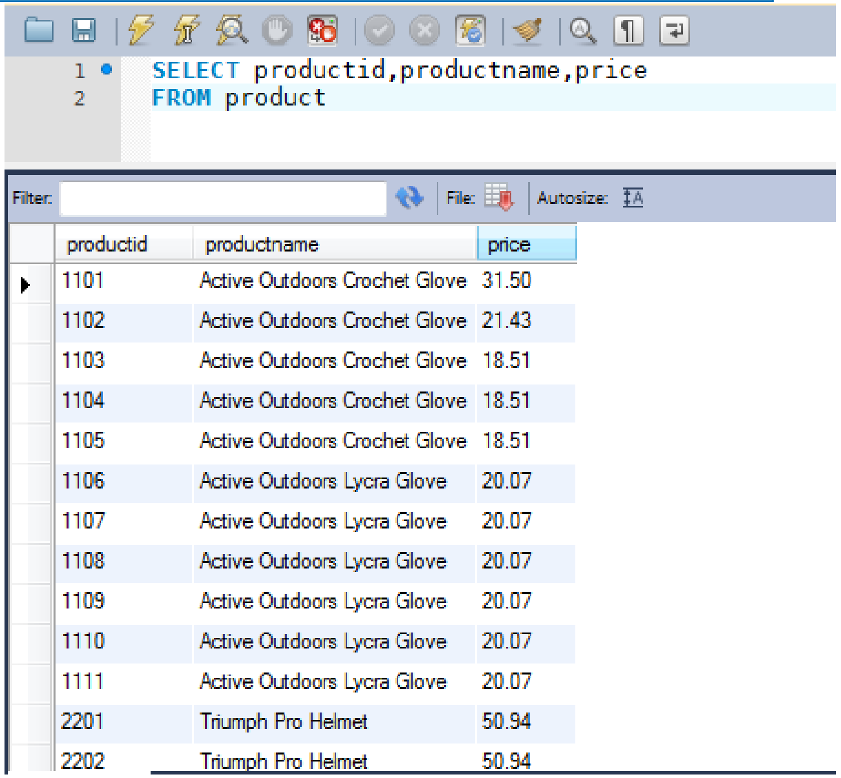
```
SELECT * FROM product
```

	ProductID	ProductName	Color	Sizes	M_F	Price	ProductTypeID	ProductClassID	SupplierID	ReorderLevel	UnitsInStock
▸	1101	Active Outdoors Crochet Glove	NULL	xsm	NULL	31.50	5	1	1	300	NULL
	1102	Active Outdoors Crochet Glove	NULL	sm	NULL	21.43	5	1	1	300	450
	1103	Active Outdoors Crochet Glove	NULL	med	NULL	18.51	5	1	1	300	325
	1104	Active Outdoors Crochet Glove	NULL	lrg	NULL	18.51	5	1	1	4	5
	1105	Active Outdoors Crochet Glove	NULL	xlrg	NULL	18.51	5	1	1	300	367
	1106	Active Outdoors Lycra Glove	NULL	xxsm	NULL	20.07	5	1	1	300	440
	1107	Active Outdoors Lycra Glove	NULL	xsm	NULL	20.07	5	1	1	300	358
	1108	Active Outdoors Lycra Glove	NULL	sm	NULL	20.07	5	1	1	300	266
	1109	Active Outdoors Lycra Glove	NULL	med	NULL	20.07	5	1	1	300	750
	1110	Active Outdoors Lycra Glove	NULL	lrg	NULL	20.07	5	1	1	300	112
	1111	Active Outdoors Lycra Glove	NULL	xlrg	NULL	20.07	5	1	1	300	686
	2201	Triumph Pro Helmet	black	sm	NULL	50.94	6	1	2	100	78
	2202	Triumph Pro Helmet	white	sm	NULL	50.94	6	1	2	100	77

SELECT

- example 2: show for all a products productID, name and unit price

```
SELECT productid,productname,price
FROM product
```



```
SELECT productid,productname,price
FROM product
```

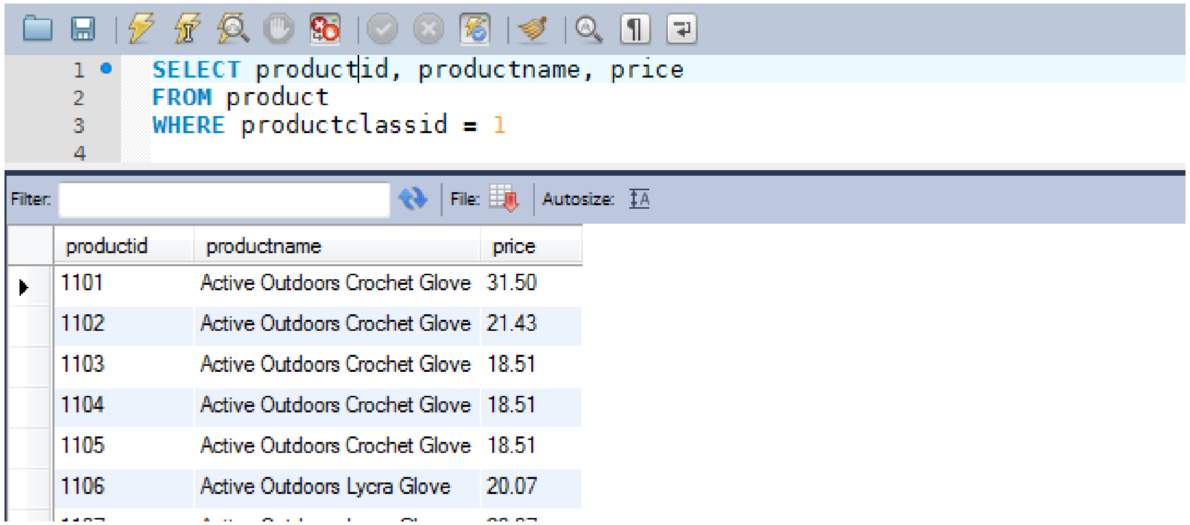
Filter: File: Autosize:

productid	productname	price
1101	Active Outdoors Crochet Glove	31.50
1102	Active Outdoors Crochet Glove	21.43
1103	Active Outdoors Crochet Glove	18.51
1104	Active Outdoors Crochet Glove	18.51
1105	Active Outdoors Crochet Glove	18.51
1106	Active Outdoors Lycra Glove	20.07
1107	Active Outdoors Lycra Glove	20.07
1108	Active Outdoors Lycra Glove	20.07
1109	Active Outdoors Lycra Glove	20.07
1110	Active Outdoors Lycra Glove	20.07
1111	Active Outdoors Lycra Glove	20.07
2201	Triumph Pro Helmet	50.94
2202	Triumph Pro Helmet	50.94

SELECT … WHERE

- WHERE clause
  - Specification of conditions for individual row
- example: show productID, name and unit price of all products from product class 1

```
SELECT productid, productname, price
FROM product
WHERE productclassid = 1
```



```
SELECT productid, productname, price
FROM product
WHERE productclassid = 1
```

Filter: File: Autosize:

productid	productname	price
1101	Active Outdoors Crochet Glove	31.50
1102	Active Outdoors Crochet Glove	21.43
1103	Active Outdoors Crochet Glove	18.51
1104	Active Outdoors Crochet Glove	18.51
1105	Active Outdoors Crochet Glove	18.51
1106	Active Outdoors Lycra Glove	20.07

SELECT … WHERE

- Use of literals
  - Numeric values: ... WHERE categoryID = 1
  - Alphanumeric values: ... WHERE productName = 'Chai'
  - Dates: ... WHERE orderDate = '4/15/1998' (15th april 1998)
- Conditions for rows
  - Comparison operators
  - Wildcards
  - Logical operators
  - Interval of specific values
  - List of values
  - Unknown values
  - Use brackets () to overrule priority rules and enhance readability

SELECT … WHERE

- Comparison operators
  - **=, >, >=, <, <=, <>**
  - Examples
    - show productID, name, units in stock for all products with less than 5 units in stock

```
select productid, productname, unitsinstock
from product
where unitsinstock < 5
```

    - show productID, name, units in stock for all products for which the name starts with A

```
select productid, productname, unitsinstock
from product
where productname >=  'A' and
      productname < 'B'
```

SELECT … WHERE

- Wildcards (searching for patterns)
  - Always in combination with operator **LIKE, NOT LIKE**
  - Wildcard symbols:
    - % arbitrary sequence of 0, 1 or more characters
    - _ 1 character
    - [ ] 1 character in a specified range
    - [^] every character not in the specified range
  - Example: show productID, name of the products for which the second letter is in the range a-k

```
SELECT productid, productname
FROM product
WHERE productname LIKE '_[a-k]%'
```

SELECT … WHERE

- Logical operators
  - **OR, AND, NOT** (ascending priority)
  - Example

```
SELECT productid, productname, supplierid, price
FROM product
WHERE (productname LIKE 'T%' OR productid = 46) AND price > 16.00
```

```
SELECT productid, productname, supplierid, price
FROM product
WHERE productname LIKE 'T%' OR (productid = 46 AND price > 16.00)
```

SELECT … WHERE

- Values in an interval
  - **BETWEEN, NOT BETWEEN**
  - Example: select the products (name and unit price) for which the unit price is between 10 and 15 euro (boundaries included)

```
SELECT productid, price
FROM product
WHERE price BETWEEN 10 AND 15
```

SELECT … WHERE

- List of values
  - **IN, NOT IN**
  - Example: show productID, name, supplierID of the products supplied by suppliers with ID 1, 3 of 5

```
SELECT productid, productname, supplierid
FROM product
WHERE supplierid in (1,3,5)
```

SELECT … WHERE

- Test for unknow (or empty) values
  - **IS NULL, IS NOT NULL**
    - NULL values occur if no value has been specified for a column when creating a record
    - A NULL is not equal to 0 (for numerical values), blank or empty string (for character values)!
    - NULL fields or considered as eqal (for e.g. testing with DISTINCT)
    - If a NULL values appears in an expression the result is always NULL/ 10 * NULL → NULL
  - Example: Select suppliers from an unknown region

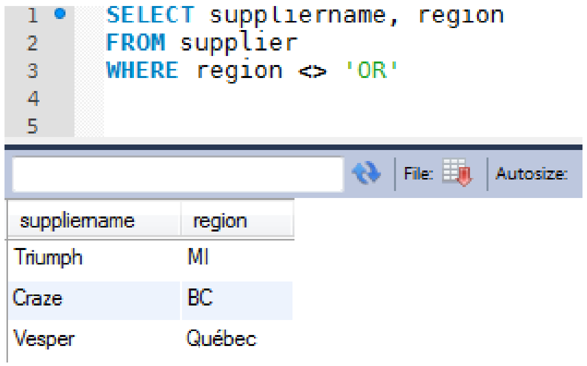
```
SELECT suppliername, region
FROM supplier
WHERE region IS NULL
```

SELECT … WHERE

- Be careful with **NULL**!

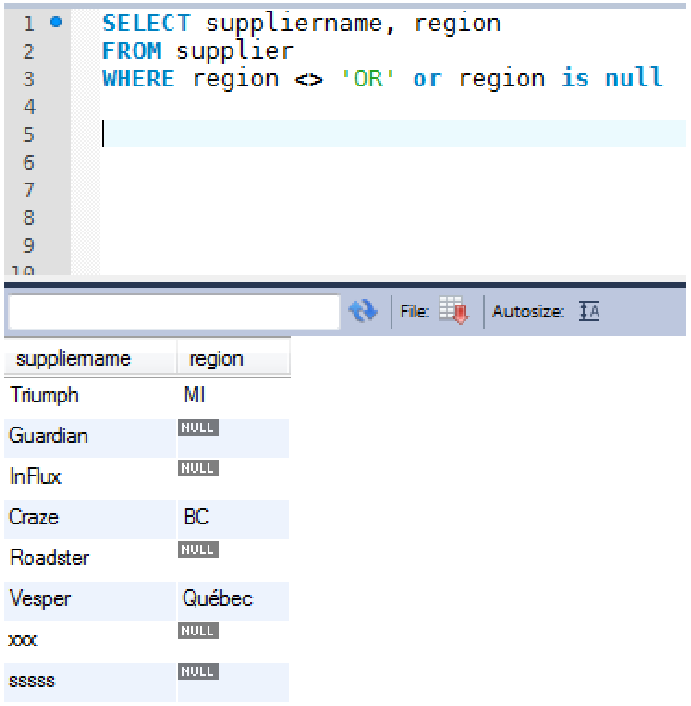
```
SELECT suppliername, region
FROM supplier
WHERE region <> 'OR'
```

```
SELECT suppliername, region
FROM supplier
WHERE region <> 'OR' OR region IS NULL
```



```
SELECT suppliername, region
FROM supplier
WHERE region <> 'OR'
```

suppliername	region
Triumph	MI
Craze	BC
Vesper	Québec



```
SELECT suppliername, region
FROM supplier
WHERE region <> 'OR' or region is null
```

suppliername	region
Triumph	MI
Guardian	NULL
InFlux	NULL
Craze	BC
Roadster	NULL
Vesper	Québec
xxx	NULL
sssss	NULL

HoGent

SELECT + formatting results

- Sorting data
- Elimination of duplicates
- Change column name in output
- Calculated output columns
- Comments
  - /* comments */
  - -- comments (rest of line is comment)

SELECT … ORDER BY

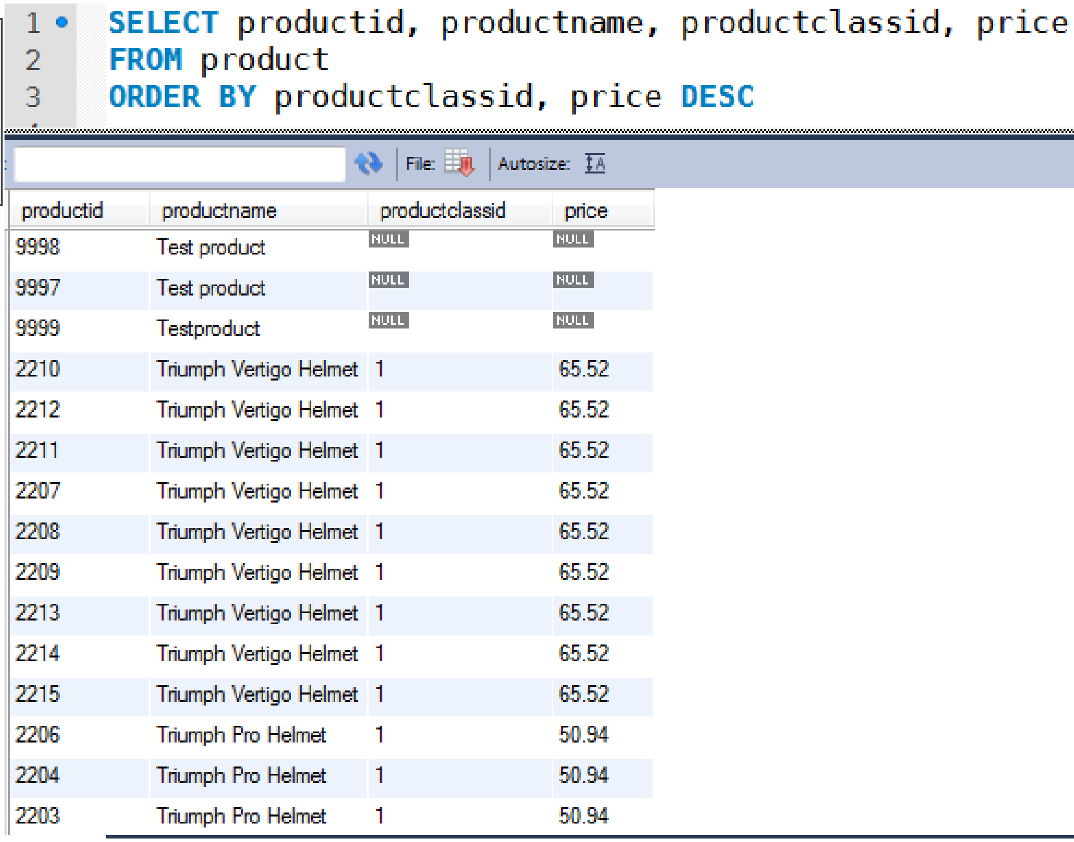
- Sorting of data
  - **ORDER BY** clause
    - Sorting according to one or more sorting criteria
    - Each sorting criterion can be specified by either a column name, an expression or a sequence number that corresponds to the order of columns in the SELECT clause (starting from 1)
    - Sorting criteria are evaluated left to right
    - Default sort occurs in ascending order (ASC: default), if descending order is required specify DESC after the criterion
  - Example: show an alphabetic list of product names

```
SELECT productname
FROM product
ORDER BY productname      -- or ORDER BY 1
```

SELECT … ORDER BY

- Example: show productid, name, productclassid of the products sorted by productclassid. If the class is the same products with the highest price appear first.

```
SELECT productid, productname,
productclassid, price
FROM product
ORDER BY productclassid, price DESC
```



```
SELECT productid, productname, productclassid, price
FROM product
ORDER BY productclassid, price DESC
```

productid	productname	productclassid	price
9998	Test product	NULL	NULL
9997	Test product	NULL	NULL
9999	Testproduct	NULL	NULL
2210	Triumph Vertigo Helmet	1	65.52
2212	Triumph Vertigo Helmet	1	65.52
2211	Triumph Vertigo Helmet	1	65.52
2207	Triumph Vertigo Helmet	1	65.52
2208	Triumph Vertigo Helmet	1	65.52
2209	Triumph Vertigo Helmet	1	65.52
2213	Triumph Vertigo Helmet	1	65.52
2214	Triumph Vertigo Helmet	1	65.52
2215	Triumph Vertigo Helmet	1	65.52
2206	Triumph Pro Helmet	1	50.94
2204	Triumph Pro Helmet	1	50.94
2203	Triumph Pro Helmet	1	50.94

SELECT DISTINCT/ALL

- Uniqueness of rows
- **DISTINCT** filters out duplicates lines in the output
  - **ALL** (default) shows all rows, including duplicates
  - Example: show all suppliers that supply products

```
SELECT supplierid
FROM product
ORDER BY supplierid
```

```
SELECT DISTINCT
supplierid
FROM product
ORDER BY supplierid
```

Some exercises

Database Xtreme:

1. Give the names of all products containing the word 'helmet' or with a name of 6 characters.
2. Show the name and the reorderlevel of all products with a level between 50 and 500 (boundaries included)

SELECT and aliasses

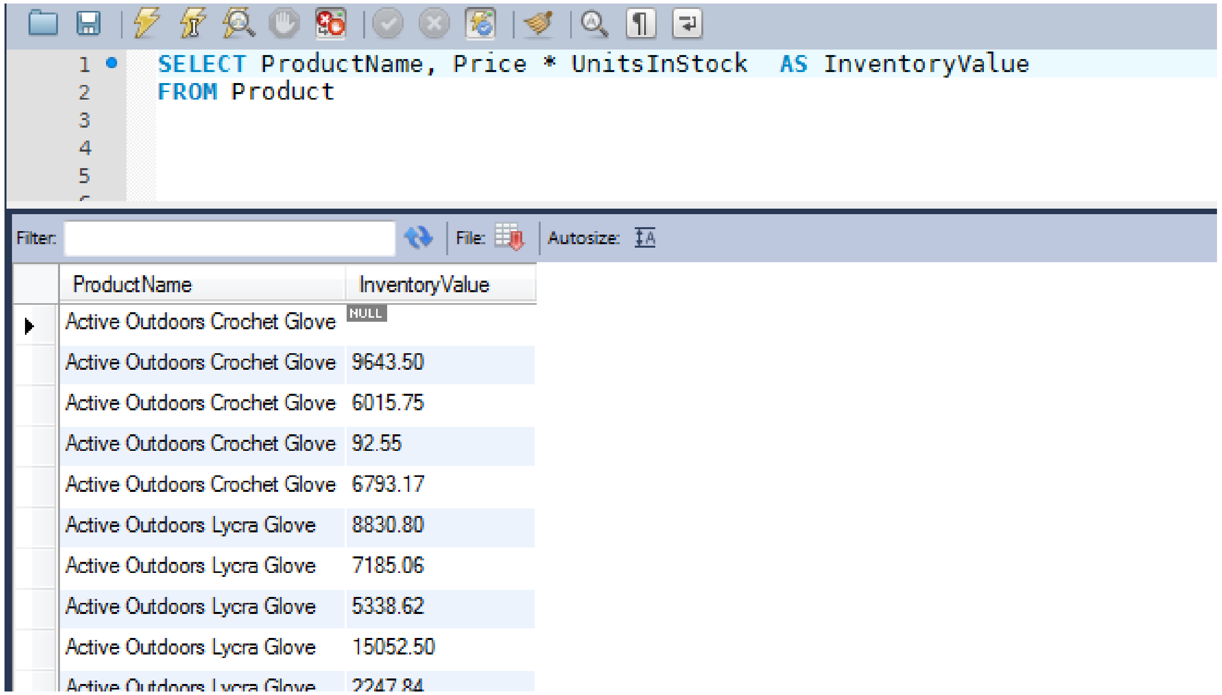
- Column names in output
  - Default : column title = name of column in table; calculated columns are unnamed
  - The **AS** keyword allows you to give a column a new title
    - Remark: the new column name can only be used in ORDER BY (not in WHERE, HAVING, GROUP BY)
  - Example: select ProductID, ProductName of the products:

```
SELECT productid AS ProductNummer,
            productname AS “Name Product”
FROM product
```

SELECT with calculated results

- Calculated result columns
  - Arithmetic operators : +, -, /, *
  - Example : give name and inventory value of the products

```
SELECT ProductName, Price * UnitsInStock  AS InventoryValue
FROM Product
```



```
SELECT ProductName, Price * UnitsInStock  AS InventoryValue
FROM Product
```

ProductName	InventoryValue
Active Outdoors Crochet Glove	NULL
Active Outdoors Crochet Glove	9643.50
Active Outdoors Crochet Glove	6015.75
Active Outdoors Crochet Glove	92.55
Active Outdoors Crochet Glove	6793.17
Active Outdoors Lycra Glove	8830.80
Active Outdoors Lycra Glove	7185.06
Active Outdoors Lycra Glove	5338.62
Active Outdoors Lycra Glove	15052.50
Active Outdoors Lycra Glove	2247.84

HoGent

SELECT and use of functions

- Functies
  - **String** functions: left, right, len, ltrim, rtrim, substring, replace, ...
  - **DateTime** functions: DateAdd, DateDiff, DatePart, Day, Month, Year, ...
    - GETDATE(): returns current date and time in DATETIME format specified by MS-SQL Server.
  - **Arithmetic** functions : round, floor, ceiling, cos, sin, ...
  - **Aggregate** functions : AVG, SUM, ...
  - **ISNULL** : replaces NULL values with specified value Example: SELECT ISNULL(unitprice, 10.00) FROM products
  - Reference document: http://msdn.microsoft.com/en-us/library/ms174318.aspx

SELECT and data type conversion

- Implicit conversions
  - Sometimes possible
  - Example: Unitsinstock * 0.5
    UnitInStock (int) is automatically converted to decimal
- Explicit conversions
  - **CAST** (<value expression> AS <data type>)
  - Example: PRINT CAST(-25.25 AS INTEGER) -> -25

  - **CONVERT** (<data type, <expression> [, <style>])
  - Example: CONVERT(VARCHAR, getdate(), 106) -> 25 nov 2014

  -
    ```
    select * from orders where
    format(shipdate,'dd/MM/yyyy')='02/12/2001';
    ```

String functions

		SQL SERVER

concatenate		SELECT concat(address,' ',city) FROM employee
		SELECT address + ' ' + city FROM employee;

substring		SELECT SUBSTRING(address,1,5) FROM employee;
Left part		SELECT LEFT(address,5) FROM employee;
Right part		SELECT RIGHT(address,5) FROM employee;
length		SELECT LEN(address) FROM employee
lowercase		SELECT LOWER(address) FROM employee
uppercase		SELECT UPPER(address) FROM employee
Remove spaces left and right		SELECT RTRIM(LTRIM(address)) FROM employee

HoGent

Date / time functions

		SQL SERVER
System date		SELECT GETDATE()
Add years, months, days to date		DATEADD (year, 2, GETDATE()) DATEADD (month, 2, GETDATE()) DATEADD (day, 2, GETDATE())
Number of years, months, days between 2 dates		SELECT DATEDIFF(day,BIRTHDATE,GETDATE()) FROM EMPLOYEE;
Day of the month		DAY(GETDATE())
Month of the year		MONTH(GETDATE())
Year		YEAR(GETDATE())

HoGent

Arithmetic functions

		SQL SERVER
absolute value		ABS(-10) → 10
Round to give number of decimals		ROUND(**10.75**,**1**) → 10.8
Largest integer thas is lower		FLOOR(10.75) → 10
Smallest integer that is higher		CEILING(**10.75**) → 11

HoGent

The case function

- Simple CASE expression:

```
select case region
when 'OR' then 'West'
when 'MI' then 'North'
else 'Elsewhere'
end,city,region
from supplier;
```

The case function

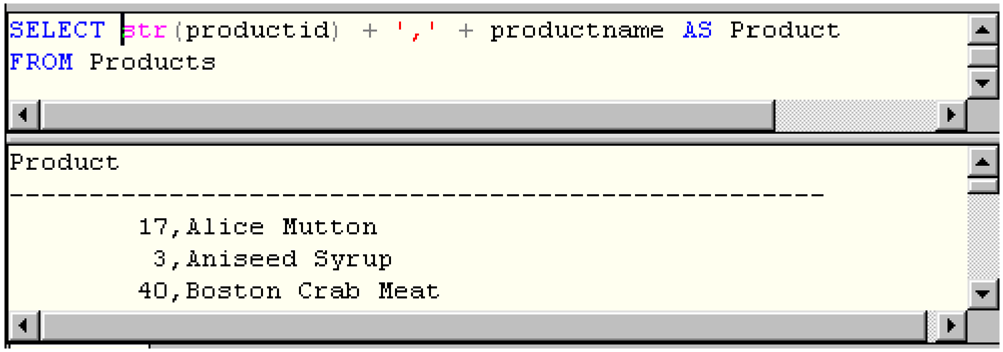
- Searched CASE expression:

```
SELECT
   CASE
      WHEN price IS NULL THEN 'Not yet priced'
      WHEN price < 10 THEN 'Very Reasonable Price'
      WHEN price >= 10 and price < 20 THEN 'Affordable'
      ELSE 'Expensive!'
   END AS "Price Category",
CONVERT(varchar(20), productname)
FROM product
ORDER BY price
```

SELECT and strings

- String operator: concatenate

```
SELECT STR(productid) + ','+ productname AS Product
FROM Product
```

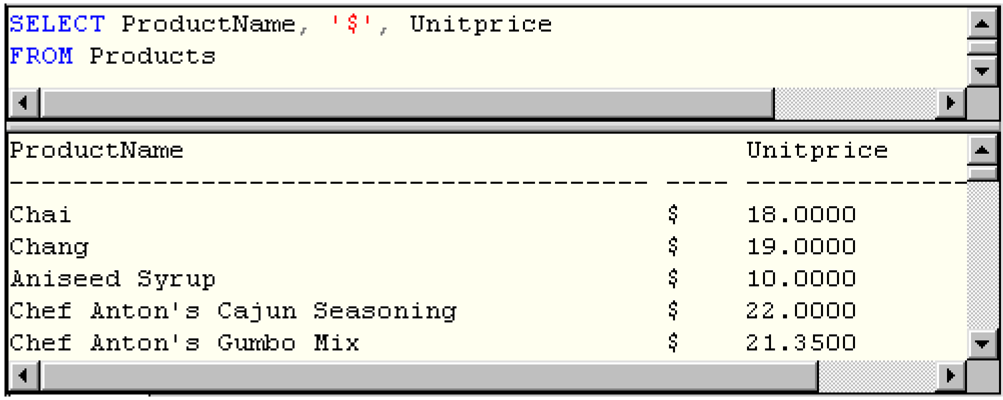


```
SELECT str(productid) + ',' + productname AS Product
FROM Products
```

```
Product
--------------------------------------------------
        17,Alice Mutton
         3,Aniseed Syrup
        40,Boston Crab Meat
```

- Use of literal text (literals)

```
SELECT  ProductName,
'$',Unitprice
FROM Product
```



```
SELECT ProductName, '$', Unitprice
FROM Products
```

```
ProductName                                   Unitprice
-------------------------------------- ---- -------------
Chai                                    $     18.0000
Chang                                   $     19.0000
Aniseed Syrup                           $     10.0000
Chef Anton's Cajun Seasoning            $     22.0000
Chef Anton's Gumbo Mix                  $     21.3500
```

HoGent

GROUP BY and statististical functions

HoGent

statistical functions

- Statistical functions (aka aggregate functions)
  - SQL has 5 standard functions
    - **SUM** (expression): sum
    - **AVG** (expression): average
    - **MIN** (expression): minimum
    - **MAX** (expression): maximum
    - **COUNT (*|[DISTINCT] column name)**: count
  - These functions give one answer per column (or group: see further) and can never be used in a where-clause

sum and average

- SUM
  - Returns the sum of all (numeric) values in a column
  - Can only be used with numeric columns
  - Example: Give the total stock value

```
SELECT SUM(UnitsInStock * Price) as inventoryvalue
FROM product
```

- AVG
  - Returns the average of <u>NOT NULL</u> numeric values in a columns
  - Can only be used with numeric columns
  - Example: What is the average number of products in stock?

```
SELECT AVG(unitsinstock) AS AverageStock
FROM product
```

Count the number of rows

- COUNT
  - Returns the number of rows, or a number of NOT NULL values in a column
    - **COUNT(*)** – counts the number of rows in a SELECT
    - Example: count the number of products (= all rows)

```
SELECT COUNT(*) as Number
FROM product
```

- **COUNT (column name)** – counts the number of not empty fields in a column
- Example: count the number of NOT NULL values in column producttypeid

```
SELECT COUNT(producttypeid) as type_count
FROM product
```

- **COUNT(DISTINCT column name)** - count the number of different NOT NULL values in column producttypeid
- Example: count the number of different NOT NULL values in column producttypeid

```
SELECT COUNT(DISTINCT(producttypeid)) as type_count FROM
product
```

minimum and maximum

- MIN and MAX
  - Returns the smallest and largest value in a column
  - Applicable for both numeric, alphanumeric and datetime fields
  - Example: what is the cheapest and most expensive unit price?

```
SELECT MIN(price) AS Minimum,
              MAX(price) AS Maximum
FROM product
```

Remark

Since a statistical function returns only **1 result**, either all expressions in the SELECT clause have to contain a statistical function, or none! This is slightly different if you use grouping (see further).

Statistical functions do not take into account **NULL values**.
Exception : COUNT(*) also counts rows with NULL values.

Transact-SQL dialect

- Some statistical functions only exists in MS Transact-SQL
  - STDEV: standard deviation of column values
  - VAR: variance of column values
  - TOP:
    - Select the top 5 of the cheapest products

```
SELECT TOP 5 productid, unitprice
FROM products
ORDER BY unitprice
```

    - 5 most expensive products: ORDER BY unitprice DESC

Grouping with GROUP BY

- Grouping – Statistical functions per group.
  - **GROUP BY** clause :
    - The table is divided into groups of rows with common characteristics.
    - Per group one unique row!
    - Example: what are the producttypes of the products?

```
SELECT ProductTypeID
FROM Product
GROUP BY ProductTypeID
```

    - For each group statistical functions can be applied.
    - The column names (or grouping criteria) mentioned in the GROUP BY clause can also appear in the SELECT clause

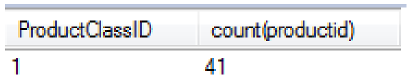
Grouping with GROUP BY

- Some examples
  - Show per type the number of products

```
SELECT ProductTypeID,count(productid)
FROM Product
GROUP BY ProductTypeID
```

  - Show per type the number of products that have more than 10 items in stock.

```
SELECT ProductTypeID,count(productid)
FROM Product
WHERE unitsinstock > 10
GROUP BY ProductTypeID
```



ProductClassID	count(productid)
1	41

Filter on groups with HAVING

- **HAVING** clause
  - Select or reject groups based on group characteristics
  - Some examples:
    - Show per type that contains more than 10 products the number of products

```
SELECT ProductTypeID,count(productid)
FROM Product
GROUP BY ProductTypeID
HAVING COUNT(PRODUCTID) > 10
```

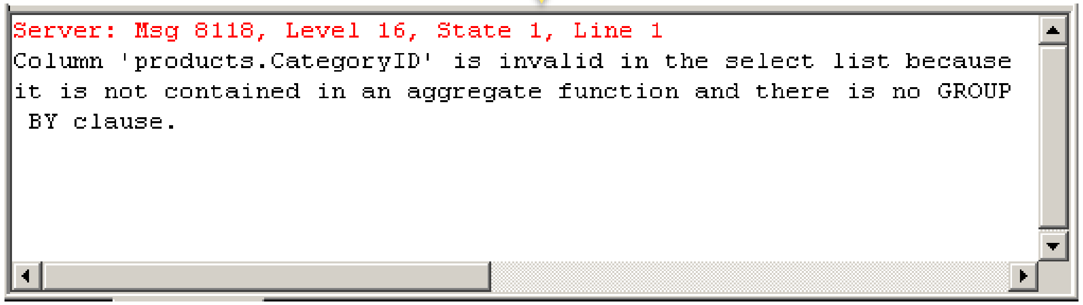
    - Show per type that contains more than 10 products with more than 10 units in stock the number of products

```
SELECT ProductTypeID,count(productid)
FROM Product
WHERE unitsinstock > 10
GROUP BY ProductTypeID
HAVING COUNT(PRODUCTID) > 10
```

WHERE vs HAVING

- Remarks
  - WHERE vs HAVING
    - WHERE – works on individual rows
    - HAVING – works on groups
  - Statististical functions can only be used in SELECT, HAVING, ORDER BY, not in WHERE, GROUP BY
  - If statistical functions appear in the SELECT, then all items in the SELECT-list have to be either statistical functions or group identifications

```
SELECT categoryID, MIN(price) AS Minimum
FROM product
```



```
Server: Msg 8118, Level 16, State 1, Line 1
Column 'products.CategoryID' is invalid in the select list because
it is not contained in an aggregate function and there is no GROUP
 BY clause.
```

HoGent

Some exercises

Database Xtreme:

1. Count the amount of products (columnname ‘amount of products’), AND the amount of products in stock (= unitsinstock not empty) (columnname ‘Units in stock’)
2. How many employees have a function of supervisor?
3. Give the date of birth of the youngest employee and the oldest.
4. What’s the number of employees who will retire (at 65) within the first 30 years?
5. Show a list of different countries where 2 of more suppliers live in. Order alphabetical.
6. Which suppliers offer at least 10 products with a price less then 100 dollar? Show supplierId and the number of different products. The supplier with the highest number of products comes first.
7. Count he number of workers (salary below 40000), clerks (salary between 40000 and 50000) and managers (salary > 50000)

Database tennis:

8. In which towns live more than 5 players?